***Commands:***

|  |  |
| --- | --- |
| **Command** | **Description** |
| **free -m** | To check ram usage. |
| **uptime** | To check the uptime of server. |
| **ANSIBLE\_KEEP\_REMOTE\_FILES=1** | To keep ansible modules in remote nodes. |
| **ansible db -m copy -a “src=/filepath dest=/destlocation”** | To copy file from ansible engine to remote node |
| **ansible db -m file -a “path=/tmp/newfile.txt state=touch mode=’0777’”** | To create a file in ansible node |
| **ansible db -m file -a “path=/tmp/newfile.txt state=absent”** | To delete a file in ansible node |
| **ansible db -m yum -a “name=git state=present” -b** | To install a package using root privileges |
| **ansible db -m yum -a “name=git state=absent” -b** | To uninstall a package |
| **ansible db -m yum -a “name=git state=latest” -b** | To upgrade a package that’s already existing |
| **ansible db -m command -a “ls> myrest.txt”** | To use command module |
| **ansible db -m setup -a “filter=ansible\_local”** | To get custom facts from ansible node |
| **Gather\_Facts:** no | Will stop collecting facts |
| **Become: yes** | To run as root user |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Ansible control node is supported by only Linux and its not supported by windows

Ansible uses SSH connection to work with worker nodes

* Password Authentication
* Password-less Authentication

Create same user ansadmin across all servers and provide passwords for all users

Provide root previliges to all ansadmin users on all servers

ansadmin ALL=(ALL) NOPASSWD: ALL

Make sure password authentication is yes for all servers

/etc/ssh/sshd\_config file

systemctl restart sshd

Generate ssh keys in the ansible control node

Using ansadmin user generate the keys using ssh-keygen.

Go to .ssh folder and copy the keys

ssh-copy-id <hostname>

***Ansible Directory***

/etc/ansible – Home directory of ansible where all configurations reside.

ansible.cfg – Incase you change your inventory file location you must mention in ansible.cfg where in the it has parameter to store inventory location. By default, hosts are where the inventory file is located.

***Ansible Ad-Hoc Commands***

One liner command that gets executed on remote node for a single task.

Module is nothing but a python script that gets executed in all managed nodes.

ansible.cfg has forks value. And if forks value is 5 and you have 12 servers , it means 5 servers it gets executed parallelly in one go and in next go it executes next 5 and finally 2 in third go.

Ansible follows push mechanism. Chef follows pull mechanism. Ansible is agent less architecture.

If you want to make it as a serial execution keep the forks value to 0.

Transfer files from ansible to remote using COPY module.

***Transfer file from ansible engine to node: (Copy Module)***

ansible db -m copy -a “src=/filepath dest=/destlocation”

Ansible is idempotent: If there is any change, then only it’s going to copy or else it won’t change.

***ansible db -m copy -a "content='I am god\n' dest='/tmp/god.txt' backup=yes"*** – To copy content into a single file and have backup available.

***ansible db -m copy -a "src='/etc/hosts' dest='/tmp'"*** – to copy a directory

***Download a file from Node to Ansible Engine***

ansible db\_servers -m fetch -a “src=/source/file/path dest=/dest/location flat=yes”

***Create and delete file/directory using ansible ad-hoc command***

ansible db -m file -a “path=/tmp/newfile.txt state=touch mode=’0777’” – To create a file

ansible db -m file -a “path=/tmp/newfile.txt state=absent” – To delete a file

ansible db -m file -a “path=/etc/demo.txt state=touch”

**If you want to run an ansible command with root privileges, you should give -b or –become=yes**

Generic ansible commands

ansible [-i inventory\_file] <server:groups:grups> -m <module> [-a arguments]

***Install a package using ansible ad-hoc commands***

State= present – to install freshly

State= absent – to remove package

State= latest – to upgrade to latest version

ansible db -m yum -a “name=git state=latest” -b – we need to be root to install any package

***Command Module***

On command line default module is shell

Command module will not use shell module instead it uses python script, so some environment variables like $HOME etc are not available.

ansible db -m command -a “ls> myrest.txt”.

***Ansible Facts***

* Information about your managed nodes. OS distribution, release, processor, python etc…
* The task of collecting this remote system information is called gathering facts and collected gathered information is called facts or variable
* You can gather facts using setup module on command line usually in playbook facts are collected by default.

***Setup Module***

* The task of collecting remote system information is called gathering facts. Gathered information is called facts or variables.
* Managed node information can be obtained using **setup** module.
* Ansible playbooks by default will run the setup module.

ansible group\_name -m setup -a “filter=ansible\_mounts”

ansible group\_name -m setup -a “filter=ansible\_architecture”

Two types of facts:

* Default facts
* Custom Facts

To get user defined required facts.

Ex: I need to find the git version, httpd version, weblogic version, db version etc.. for all managed nodes

ansible all -m shell -a “httpd version”

We can create some facts on managed nodes, and we can get them using setup module. Custom facts are nothing but required values ( git version, httpd version etc…..)

***Step1:*** /etc/ansible/facts.d on Managed Nodes.

***Step2:*** Inside of facts.d place one or more custom fact files with extensions as .fact.

***Step 3:*** The output of fact file should be a . json.

***Step 4:*** The facts should have executable permissions.

***Step 5***: We can find custom facts under ansible\_local filter or key

**ansible db -m setup -a “filter=ansible\_local”**

***Ansible Inventories***

The collection of hosts is called ansible inventory.

There are two types of inventories:

* Static Inventory
* Dynamic Inventory
* Generally dynamic inventories are scripts like shell/python for dynamic environments
* Cloud is a dynamic environment

Ansible has dynamic inventory for below environments

* AWS ec2
* OpenStack
* Google Compute Engine
* Space Walk
* Jails etc

We can mention our custom inventory using ansible -i [inventory\_name] all -m ping

**Raw Module**

If managed nodes not installed with python, we can use “RAW” module.

Working with managed nodes using password

***Password Authentication***

* Using SSH keys
* Using SSH password

***Ansible Variables***

Ansible variables are letters, numbers, and underscores and they should always start with a letter.

Types of variables

Default Variables

Inventory Vars

Facts and Local Facts

Registered vars

***Default Variables***

* inventory\_hostname
* inventory\_hostname\_short
* groups/groups.keys()

ansible all -m debug -a “msg=’This is a debug module’”

debug is used to display message and it runs only on ansible engine not on managed nodes.

**ansible all -m debug -a “var=inventory\_hostname”**

**ansible all -m debug -a “msg={{inventory\_hostname}}”**

**ansible all -m debug -a “var=groups”**

***Host/Group Variables***

**ansible all -m ping**

**Two ways to execute task on Nodes**

**Playbooks**

ansible all -m ping

ansible db -m yum -a “name=httpd state=present”

Ad-hoc commands are not helpful when dealing with multiple tasks

Playbook is useful to execute multiple tasks and eliminates the limitation of ad-hoc commands

Playbook is the configuration, deployment, and orchestration language of ansible, and it’s expressed in yaml format

* **task**
* **play**
* **playbook**

Task: Installing httpd, Installing java

Collection of tasks with target servers is a **play**

Collection or list or sequence of plays (One or many plays) is called **playbook**.

**Gather\_Facts:** no will stop collecting facts.

***Basic key points to run ansible playbooks***

**---**

**- name: This is a simple play on webservers**

**hosts: db**

**become: yes**

**tasks:**

**- name: This is a yum module**

**yum:**

**name: wget**

**state: absent**

**- name: This is a copy module**

**copy:**

**src: web.txt**

**dest: web.txt**

**- name: This is to create a file**

**file:**

**path: authentication.txt**

**state: touch**

***Print any message in playbook***

***Debug module***: message, variable and verbose parameters need to be passed

Debug: msg=”Welcome to ansible playbooks”

#!/usr/bin/ansible-playbook

**---**

**- name: This playbook is for checking the debug module**

**hosts: db**

**tasks:**

**- name: This is a debug task**

**debug:**

**msg: ["My name is mahesh","The inventory hostname is {{inventory\_hostname}}"]**

**verbosity: 3**

***Custom Variable***: A variable that is defined by us is called custom variable.

**#!/usr/bin/ansible-playbook**

**- name: This playbook is for defining variables**

**hosts: db**

**vars:**

**x: 56**

**y: "Mahesh"**

**z: Mamidibathula**

**k: no**

**tasks:**

**- name: This task is for showing variables**

**debug:**

**msg:**

**- "The value of x is {{x}} and type is {{x|type\_debug}}"**

**- "The value of y is {{y}} and type is {{y|type\_debug}}"**

**- "The value of z is {{z}} and type is {{z|type\_debug}}"**

**- "The value of k is {{k}} and type is {{k|type\_debug}}"**

***Sequence or list:***

***List:***

***Map or Dictionary:***

**#!/usr/bin/ansible-playbook**

**- name: This playbook is meant to define datastructures**

**hosts: db**

**vars:**

**x: {'centos':'yum','ubuntu':'apt'}**

**y: ['centos','ubuntu','debian']**

**tasks:**

**- name: This task is debug module to debug the data structures**

**debug:**

**msg:**

**- "The value of dictionary is {{x.values()}}"**

**- "The value of keys in dictionary is {{x.keys()}}"**

**- "The value of items in dictionary are {{x.items()}}"**

***Usage of register and set\_fact:***

Ansible module return a data structure that can be registered into a variable

Ansible **registers** are used to store the output of a module/task into a variable

We can use the registers for ***conditional statements***, ***logging*** and ***displaying outputs***

***set\_fact*** used to define new variables.

**#!/usr/bin/ansible-playbook**

**- name: This playbook is used to define register and set\_fact in ansible**

**hosts: db**

**tasks:**

**- name: Task to define shell module**

**shell: "bash --version"**

**register: bash\_version**

**- set\_fact:**

**bash\_version1: "{{bash\_version.stdout}}"**

**- name: This task is to get the debug messages**

**debug:**

**msg:**

**- " The value is {{bash\_version}}"**

**- " The value is {{bash\_version1}}"**

***vars\_prompt***

This is to prompt for entering a user value.

private: false, is to ensure when typing it is not displayed on screen

**#!/usr/bin/ansible-playbook**

**- name: This playbook is to define vars prompt**

**hosts: db**

**vars\_prompt:**

**- name: user\_name**

**prompt: Enter your username**

**private: no**

**- name: password**

**prompt: Enter your password**

**private: yes**

**tasks:**

**- name: This task is to see the working of debug module**

**debug:**

**msg:**

**- "The value of username is {{user\_name}}"**

**- "The value of password is {{password}}"**

***Read variable from yml and json files***

Different files to store variables

* yaml files
* Json files

vars\_files:

**#!/usr/bin/ansible-playbook**

**- name: This playbook read variables from files**

**hosts: db**

**vars\_files:**

**- varsfile.json**

**tasks:**

**- name: This task is to read values from files**

**debug:**

**var: k**

***Working with command line arguments***

We can pass different types of variables in different formats:

* Scalar
* Sequence
* Map

To pass a scalar value: -e “x=34”

-e “{‘x’:67}”

Pass multiple values:

-e “x=56 y=89”

-e “{‘x’: [34,45]}”

***Working with gather\_facts and setup module***

Working with inventory\_hostname and hostvars variables

***Comparison Operators***

* >,<,=,!=,<=,>=

***Membership operators and Test operators***

* in
* not in

**Test operators**

* Test for variables (is defined and is not defined)
* Test for strings (strx is lower, strx is upper, strx is string)
* Test for numbers ( numx is divisibleby 5,numx is even, numx is odd)
* Test for paths: {{my\_path is file}}

**Logical Operators**

* or
* and
* not

**#!/usr/bin/ansible-playbook**

**- name: This playbook is for computing logical operators**

**hosts: db**

**vars:**

**x: 56**

**p: [12,42,12,31]**

**z: 31**

**tasks:**

**- name: Task for comparison**

**debug:**

**msg:**

**- "The value of comparison {{ x in p or z in p}}"**

**- " The value of comparison {{ x in p and z in p}}"**

***Conditional Statements***

* **when: it is like if condition in other languages**
* **failed\_when**
* **changed\_when**
* **Condition/expression can be formed with comparison, membership, test and logical operators**

**#!/usr/bin/ansible-playbook**

**- name: This playbook is for installing httpd and apache2 softwares**

**hosts: db**

**become: yes**

**tasks:**

**- name: Installing webservers on managed nodes**

**yum:**

**name: httpd**

**state: present**

**when: ansible\_distribution != "Ubuntu"**

**- name: Installing apache2 server**

**apt:**

**name: apache2**

**state: present**

**when: ansible\_distribution == "Ubuntu"**

***In-line conditional statements***

It’s a way of representing if-else statements.

***inline\_condition.yml***

**#!/usr/bin/ansible-playbook**

**- name: This playbook is for showing in-line conditions**

**hosts: db**

**vars:**

**x: 56**

**y: 72**

**tasks:**

**- name: This task is to show whether x is greater than y or not**

**debug:**

**msg: "The bigger value is: {{x if x>y else y}}"**

**Handlers**

They are like regular tasks in ansible playbook.

They must be called and gets executed only when the previous task is executed.

**#!/usr/bin/ansible-playbook**

**- name: This playbook is used to describe about handlers section**

**hosts: db**

**tasks:**

**- name: This task is used to install httpd version**

**yum:**

**name: httpd**

**state: present**

**notify:**

**- httpd\_installed**

**handlers:**

**- name: httpd\_installed**

**service:**

**name: httpd**

**state: started**

**Loops**

#!/usr/bin/ansible-playbook

**- name: This playbook for loops**

**hosts: db**

**tasks:**

**- name: This task is for loops**

**debug:**

**msg: "The value is {{item}}"**

**loop: ['git','wget','httpd']**

**Tags**

Tags are useful to execute/skip required tasks from number of tasks.

--skip-tags: to skip particular tasks

-t: to execute certain tasks

--list-tags: list all the tags

- never: Tag will ensure the task is never executed

-always: It will get executed always

**Error Handling**

**ignore\_errors:yes**

This is used to pass through the next plays in playbook even the current play is failed.

**Block and rescue in playbooks**

Ansible stops playbook execution on a task failure, and we can choose to ignore errors to continue with remaining tasks

**#!/usr/bin/ansible-playbook**

**- name: This playbook is for explaining error handling**

**hosts: db**

**tasks:**

**- name: This task is for installing httpd**

**service:**

**name: httpd**

**state: started**

**ignore\_errors: yes**

**- name: nginx service**

**service:**

**name: nginx**

**state: started**

**ignore\_errors: yes**

**Error handling with block and rescue**

**At block level we can add ignore\_errors as yes so that inside any play fails it will not fail entire playbook.**

-block :

Ignore\_errors: yes

***Import and Include tasks***

Import is ***static*** and it doesn’t accept variables from ansible facts and custom facts whereas include is ***dynamic*** and it accepts ansible variables

**#!/usr/bin/ansible-playbook**

**- name: This playbook is for import and include tasks**

**gather\_facts: true**

**hosts: db**

**tasks:**

**- name: This task is for showing the import module**

**include\_tasks: install\_webserver\_{{ansible\_os\_family}}.yml**

**- name: This task is to show import module**

**import\_tasks: install\_webserver\_RedHat.yml**

**when: ansible\_os\_family=="RedHat"**

**Usage of delegate\_to, run\_once and local\_action in ansible playbooks**

Discovery script: It will generate a csv file

To redirect into a csv file, first csv file needs to be created

**#!/usr/bin/ansible-playbook**

**- name: This playbook is used for executing delegate\_to and creating files**

**gather\_facts: true**

**hosts: db**

**tasks:**

**- name: shell module**

**shell: "uname"**

**register: uname**

**- name: This module is for computing the uptime**

**shell: "uptime"**

**register: uptime**

**- name: Displaying the discovery information**

**debug:**

**msg: "{{inventory\_hostname}},{{ansible\_os\_family}},{{uname}},{{uptime}}"**

**- name: Removing old discovery .csv file**

**file:**

**dest: ./Discovery.csv**

**state: absent**

**delegate\_to: localhost**

**run\_once: true**

**- name: Create new discovery file**

**copy:**

**dest: ./Discovery.csv**

**content: "servername,osname,uname,time\n"**

**delegate\_to: localhost**

**run\_once: true**

**- name: Updating info on each server**

**lineinfile:**

**path: ./Discovery.csv**

**line: "{{inventory\_hostname}},{{ansible\_os\_family}},{{uname}},{{uptime}}"**

**delegate\_to: localhost**

delegate\_to : to execute task on one server

***location\_action***: execute task on all servers from localserver

***Install and configure tomcat using template module***

1. ***Install openjdk 1.8.0***

***Ansible-Vault:***

This is used to encrypt your playbooks

Questions:

1. How to ensure modules are not deleted on managed nodes?

Ans: ANSIBLE\_KEEP\_REMOTE\_FILES=1 option helps in ensuring the module is not deleted.

1. How does the ansible process run in remote nodes?
2. What are the steps to follow to obtain custom facts from managed nodes?
3. What is the significance of error handling and what are the ways to progress to the next plays when current play is failed?
4. What is the significance of import and include tasks and outline the differences between them?